

Discrete Fourier Transform IP Generator*

Grace Nordin James C. Hoe Markus Püeschel

Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

Abstract

Intellectual Property (IP) libraries are commonly used by hardware designers to increase productivity and reduce the time-to-market. These static IP libraries do not allow the designers flexibility in customizing trade-offs. We propose a parameterized DSP IP generator that allows designers to specify the cost/performance tradeoff. We present a prototype implementation of a parameterized DFT generator and compare our generated DFT with Xilinx LogiCore’s DFT IP Core. Our results show that we generate high-quality DFT blocks that match the performance and cost of Xilinx LogiCore DFT implementations. More importantly, we show that our parameterized design generation yields customized DFT blocks over a range of different performance/cost tradeoff points.

Introduction. We propose a parameterized IP generator as an alternative to static IP blocks. The generator is tailored for application-specific tradeoffs, such as area, performance, numerical accuracy and power consumption. Our approach preserves the advantage of using static IP blocks, while allowing the designers more control over the design. This generator can be used together with a search engine to find the best possible implementation for a given set of constraints. Here we present our experience in developing a parameterized generator for discrete Fourier transform (DFT). A full description of this work has been submitted to a conference.

Generation of Discrete Fourier Transform. Our DFT generator is based on the Pease algorithm for the DFT, which we express in a formula notation as

$$F_{2^n} = \left\{ \prod_{i=0}^{n-1} L_2^{2^n} (I_{2^{n-1}} \otimes F_2) T_{n-i} \right\} R_{2^n}, \quad F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (1)$$

where ‘I’ denotes an identity matrix, ‘ \otimes ’ the Kronecker product of matrices, ‘T’ denotes the Twiddle factors, ‘R’ denotes the bit reversal, and ‘L’ a stride permutation. Figure 1 shows a dataflow representation of (1) for $n = 3$. This formula-derived dataflow graph can be directly mapped to a combinational circuit where the implementation cost is approximately $n \log(n)/2$ C blocks, plus the routing cost of realizing the L_2^n wire permutations. The cost of a combinational implementation is usually very large and unrealistic to implement for large n . A common practical DFT implementation requires a sequential implementation where the logic resources, e.g., C , are reused multiple times by *horizontal folding* or *vertical folding*. Figure 1 shows, for $n = 3$, block diagrams of a horizontally folded DFT (middle) and a horizontally and vertically folded DFT (right).

Our DFT generator accepts as input parameters the DFT size, the data format (i.e., fixed-point number range and precision), and a design parameter p that controls the degree of parallelism in the generated implementation. This freedom allows the designer to select a custom tradeoff between minimizing cost (i.e., area and power) and maximizing performance (i.e., latency and throughput). Our DFT generator can also accept target-specific parameters to reflect the designer’s preference for different classes of resources. A parameter that our DFT generator allows is a relative value for a *Block Select-RAM* (BRAM, a specialized memory

*This work was supported by NSF through award ITR/NGS-0325687

Report Documentation Page			Form Approved OMB No. 0704-0188		
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE 01 FEB 2005		2. REPORT TYPE N/A		3. DATES COVERED -	
4. TITLE AND SUBTITLE Discrete Fourier Transform IP Generator				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release, distribution unlimited					
13. SUPPLEMENTARY NOTES See also ADM001742, HPEC-7 Volume 1, Proceedings of the Eighth Annual High Performance Embedded Computing (HPEC) Workshops, 28-30 September 2004 Volume 1. , The original document contains color images.					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 24	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

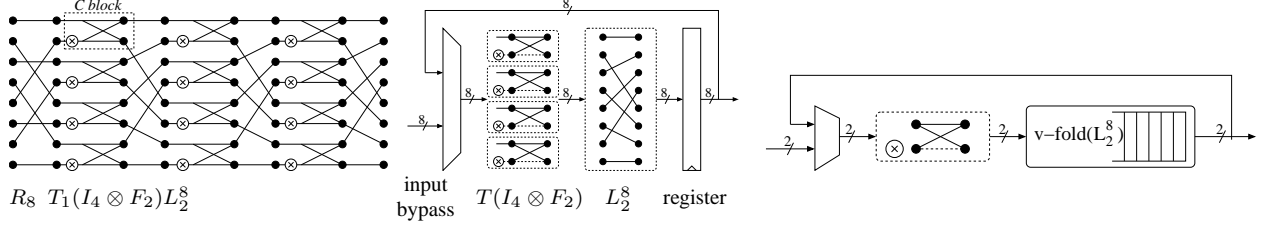


Figure 1: Pease DFT algorithm. From left to right: completely flattened, horizontally folded, fully horizontally and vertically folded.

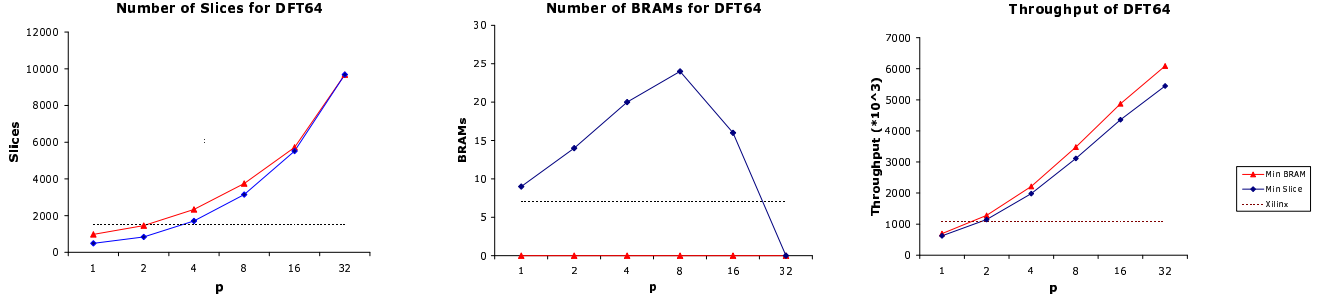


Figure 2: Synthesis results for F_{64} : slice utilization, BRAM utilization, and transform throughput (transform/second, overlapping loading and unloading).

Table 1: Parameters for DFT IP Generator and the corresponding effects on logic slices, BRAM and throughput as each parameter increases

Parameter	Logic Slices	BRAM	Throughput
n (transform size)	↑↑	↑↑	↓
p (parallelism)	↑↑	↑↑	↑
fixed-point number range and precision	↑↑	↑↑	—
relative value of BRAM cost	↓	↑	—

primitive) in terms of *slices* (generic logic building blocks). The DFT generator takes this preference into account to balance resource minimization across BRAM and logic slice utilization. Table 1 lists some current parameters that our DFT generator currently supports, and the corresponding effects on BRAM, logic slice utilization and throughput. The output of our generator is an RTL-level Verilog description of the desired DFT implementation.

Sample Result. For $n = 6$, our DFT generator produces 6 implementations representing different trade-offs between the different design goals and constraints. Figure 2 shows the resource utilization, in terms of slices and BRAM, and throughput over these 6 design choices, compared against the latest Xilinx LogiCore DFT implementations. The generated DFT implementations are synthesized for the Xilinx Virtex2-Pro XC2VP100-6FF1696 FPGA using Xilinx ISE version 6.1.03i. To show the effects of our Xilinx Virtex2-Pro-specific parameters, each graph reports two separate results corresponding to the extreme tradeoff points of slices and BRAM utilization. They are 1) minimize the use of slices; and 2) minimize the use of BRAMs. As the graphs show, our minimum resource design points (i.e., $p = 1$ or $p = 2$) occupy a similar tradeoff space as the Xilinx LogiCore DFT implementations. By varying p and the relative value of BRAM, the designer can customize the tradeoff function between performance, slice, and BRAM usage. For larger p values, our generated DFT implementations can offer a higher throughput at the cost of an increased resource requirements.

Discrete Fourier Transform IP Generator

Grace Nordin, James C. Hoe, and Markus Püschel
Dept. of Electrical and Computer Engineering
Carnegie Mellon University

The Paradox of Reusable IPs

◆ Boon to productivity

- zero effort required
- zero knowledge required
- zero chance to introduce new bugs

Why repeat what is already been done?

◆ Bane to optimality

- finding the right functionality with the right interface
- design tradeoff -- performance, area, power, accuracy.....

Are you getting what you really wanted?

◆ **Solution:** parameterized automatic IP generators

- zero effort, knowledge or bugs
- allows application specific customization
- facilitates design exploration

Discrete Fourier Transform IPs

- ◆ Discrete Fourier Transform (DFT)
 - important building block in DSP applications
 - numerous design “cores” available
- ◆ Some commonly supported options in IP libraries
 - transform sizes
 - number format
 - i/o data ordering
 - a small number of microarchitecture choices (e.g., min area, max speed)
- ◆ Customized design tradeoff in our generated IPs
 - degree of parallelism in microarchitecture (*min* \leftrightarrow *max*)
 - resource preference (e.g. *BRAM* vs. *LUT* in *FPGAs*)

Extensible to other common linear DSP transforms

Outline

- ◆ Introduction
- ◆ Formula-Driven Design Generation
- ◆ Microarchitecture Parameterization
- ◆ Resource Parameterization
- ◆ Experimental Results
- ◆ Conclusions

Transforms as Formulas [www.spiral.net]

Transform

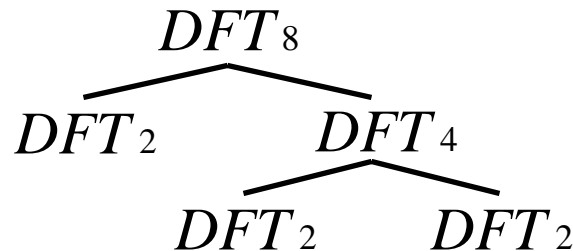
DFT_n parameterized matrix

Recursion

$$DFT_{nm} \rightarrow (DFT_n \otimes I_m) \cdot D \cdot (I_n \otimes DFT_m) \cdot P$$

- a breakdown strategy
- product of sparse matrices

Algorithm As Tree



- recursive application of rules
- uniquely defines an algorithm
- efficient representation
- easy manipulation

Algorithm As Formula

$$DFT_8 = (F_2 \otimes I_4) \cdot D \cdot (I_2 \otimes ((F_2 \otimes I_2) \cdots)) \cdot P$$

- few constructs and primitives
- uniquely defines an algorithm
- can be translated into code

Formula to Datapath

◆ Given $M \cdot x$ where M is

- $M = A \cdot B$

apply B , then A

- $M = I_n \otimes A$

apply A , n times in parallel

- $M = A \otimes I_n$

apply A , n times in parallel
taking inputs at stride n

- M is a permutation

permute x

- M is a diagonal

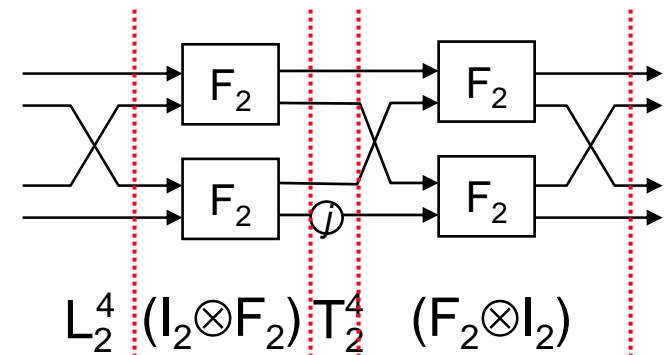
scale x

- etc.

◆ formulas are a natural HW description

◆ formulas allow manipulation

◆ formulas can be translated into Verilog



Outline

- ◆ Introduction
- ◆ Formula-Driven Design Generation
- ◆ Microarchitecture Parameterization
- ◆ Resource Parameterization
- ◆ Experimental Results
- ◆ Conclusions

Pease DFT

- ◆ Simple regular structure embodied in formula

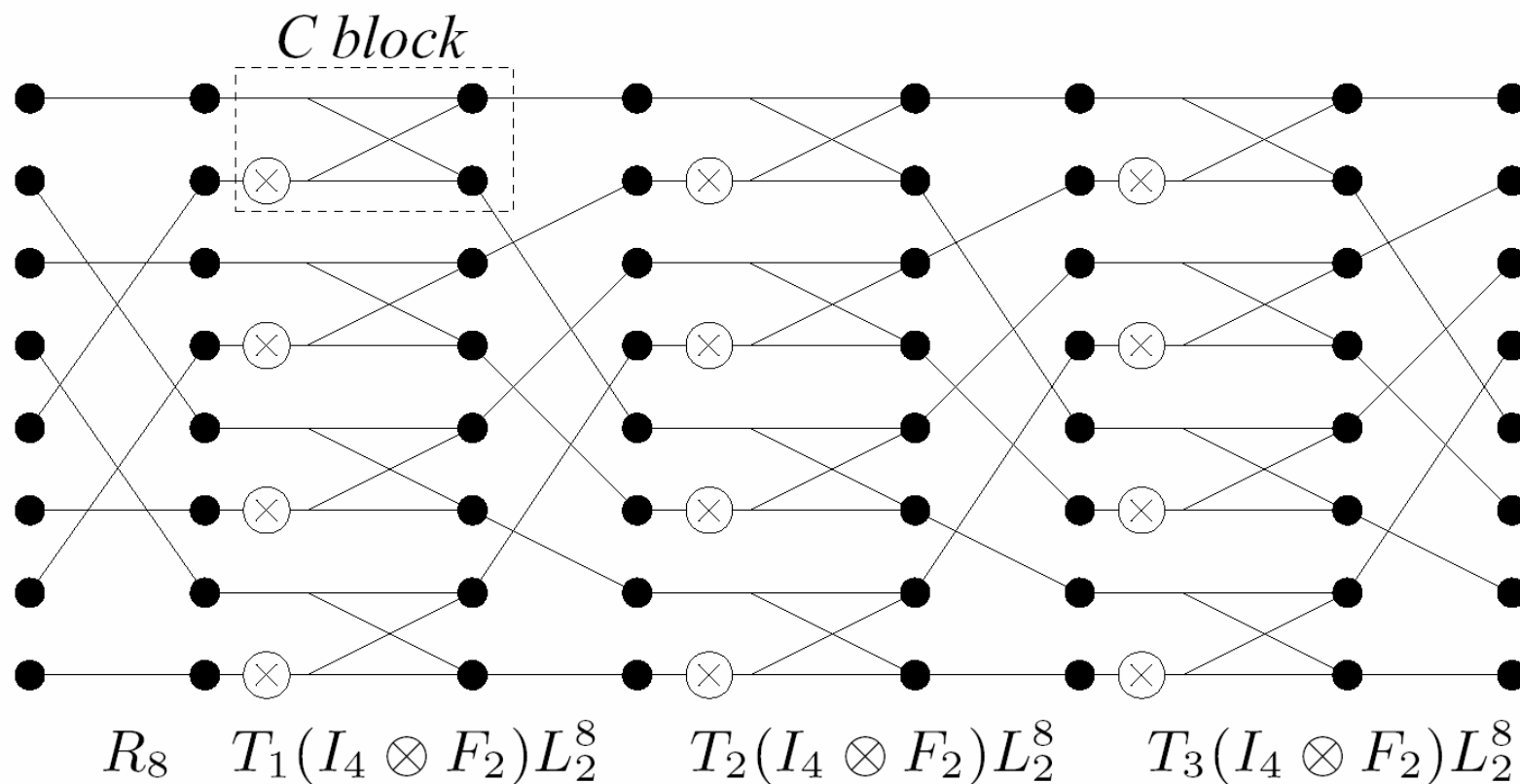
$$\text{DFT}_{2^k} = \left(\prod_{i=0}^{k-1} L_2^{2^k} (I_{2^{k-i}} \otimes F_2) T_{k-i} \right) R_{2^k}$$

$$\text{where } T_{k-i} = L_{2^{k-i-1}}^{2^k} (I_{2^i} \otimes D_{2^{k-i-1}}^{2^{k-i}}) L_{2^{i+1}}^{2^k}$$

- ◆ Example

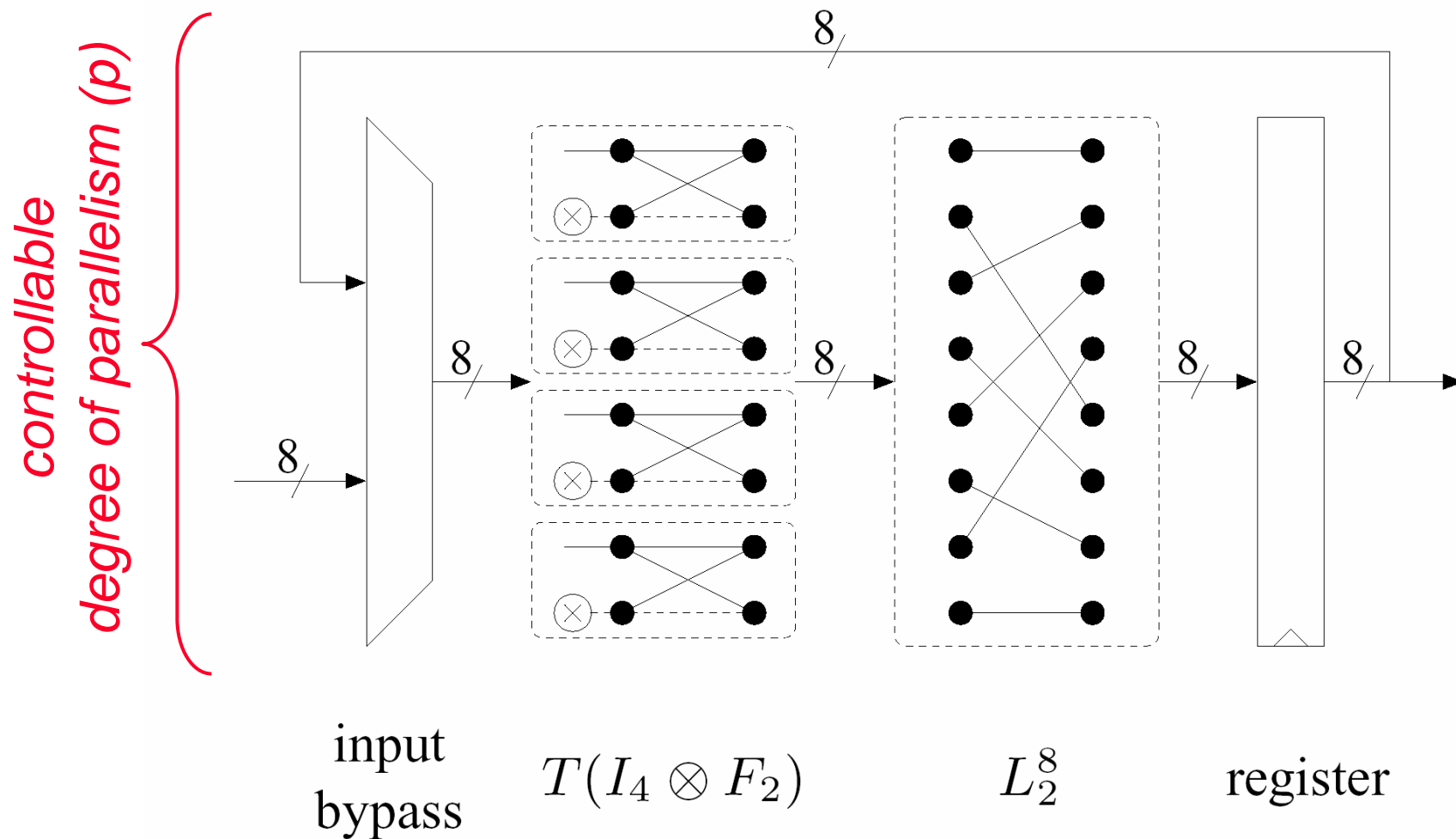
$$\begin{aligned} \text{DFT}_8 = & (L_2^8(I_4 \otimes F_2)T_3) (L_2^8(I_4 \otimes F_2)T_2) \\ & (L_2^8(I_4 \otimes F_2)T_1) R_8 \end{aligned}$$

Pease DFT Example: DFT_8

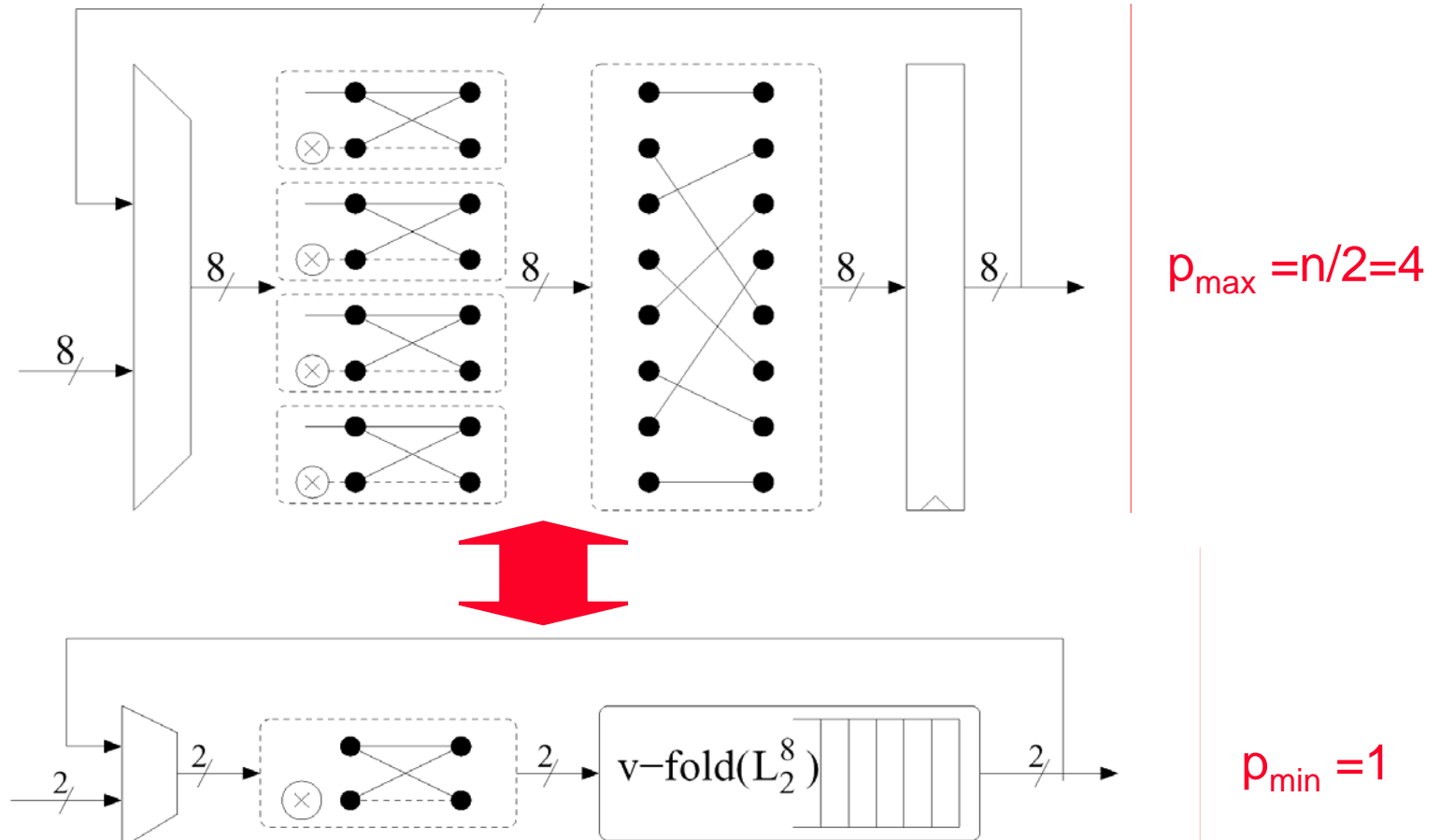


*Repeating column structure \Rightarrow hardware reuse
with zero perf. penalty*

Horizontally Folded Pease DFT



V-folding according to p



$$\text{Latency} = t \left(\frac{n}{2p} (\log(n) - 1) \right)$$

Outline

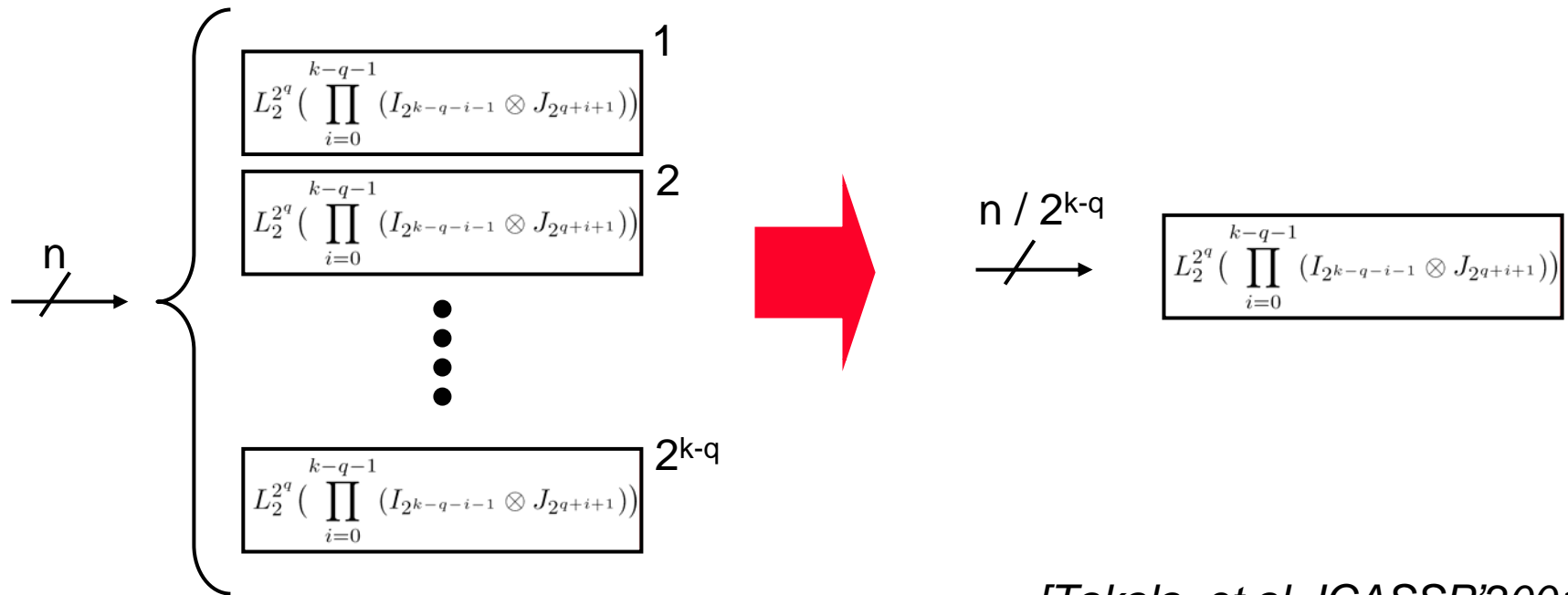
- ◆ Introduction
- ◆ Formula-Driven Design Generation
- ◆ Microarchitecture Parameterization
- ◆ Resource Parameterization
- ◆ Experimental Results
- ◆ Conclusions

V-folding of Stride Permutations

◆ Stride Permutation

$$L_2^n = I_{2^{k-q}} \otimes L_2^{2^q} \left(\prod_{i=0}^{k-q-1} (I_{2^{k-q-i-1}} \otimes J_{2^{q+i+1}}) \right)$$

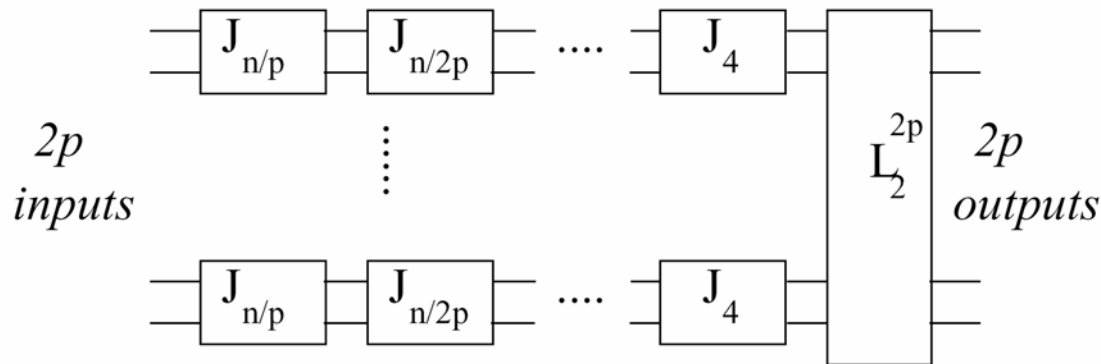
◆ In other words



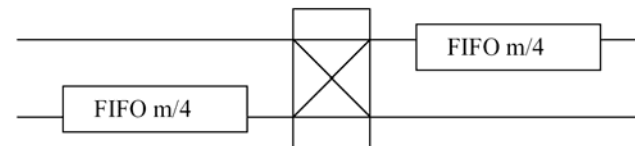
[Takala, et al. ICASSP'2001]

V-folding of Stride Permutations

$$L_2^{2^q} \left(\prod_{i=0}^{k-q-1} (I_{2^{k-q-i-1}} \otimes J_{2^{q+i+1}}) \right)$$



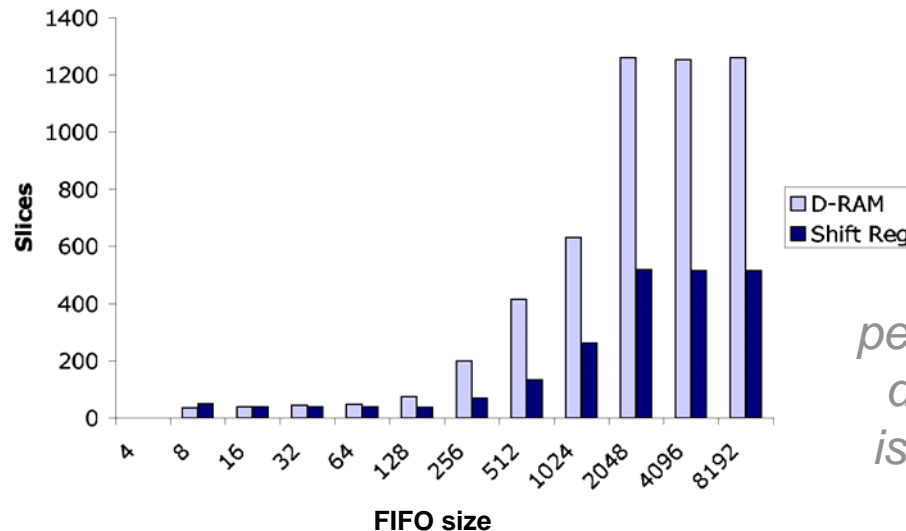
where $J_m =$



[Takala, et al. ICASSP'2001]

FIFO: BRAM vs. CLBs

- ◆ J-matrix FIFOs are a significant part of logic resources
- ◆ FIFOs can be constructed from
 - shift registers using CLB slices, or
 - circular buffers using CLB slices (distributed RAM), or
 - circular buffers using BRAM memory macros
- ◆ “Exchange rate” of shift registers vs. circular buffer



*performance
difference
is negligible*

Let user set the context-dependent break-even point

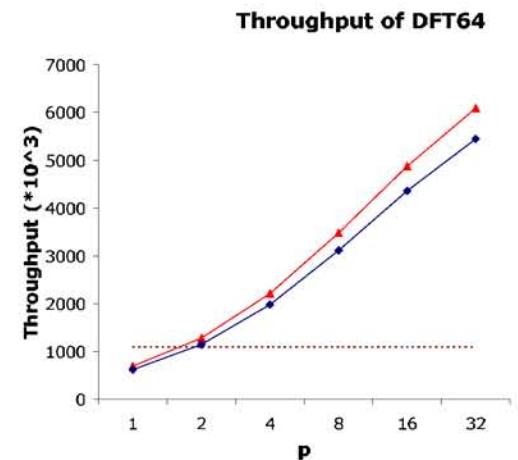
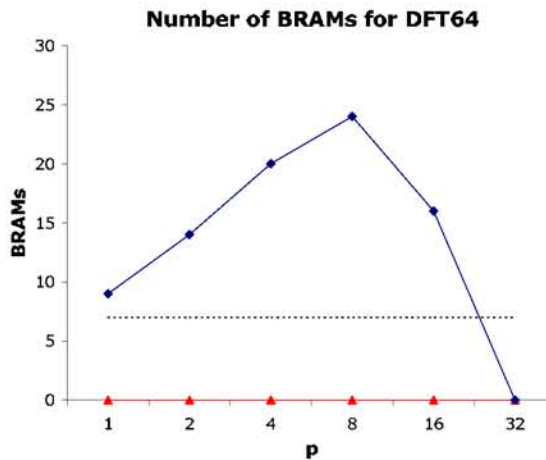
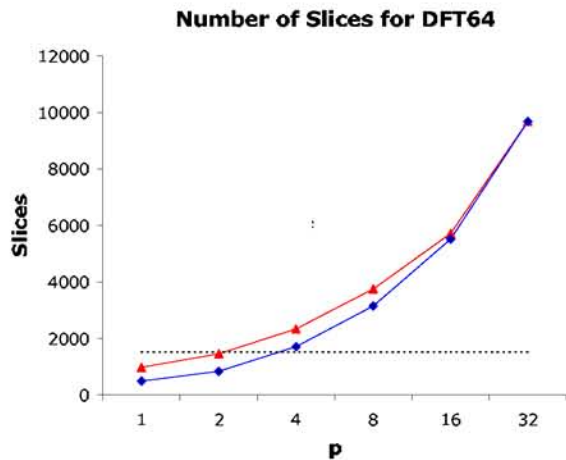
Outline

- ◆ Introduction
- ◆ Formula-Driven Design Generation
- ◆ Microarchitecture Parameterization
- ◆ Resource Parameterization
- ◆ Experimental Results
- ◆ Conclusions

Xilinx LogiCore Library

- ◆ DFT based on Radix-4 Cooley-Tukey
 - range of sizes
 - streaming vs. burst I/O
 - fixed-point scaling modes
 - in/out data ordering
- ◆ Evaluation
 - DFT of 64, 1024 and 2048
 - burst I/O interface, bit-reversed-ordering
 - Xilinx Virtex2-Pro XC2VP100-6
 - Xilinx ISE version 6.1.03i

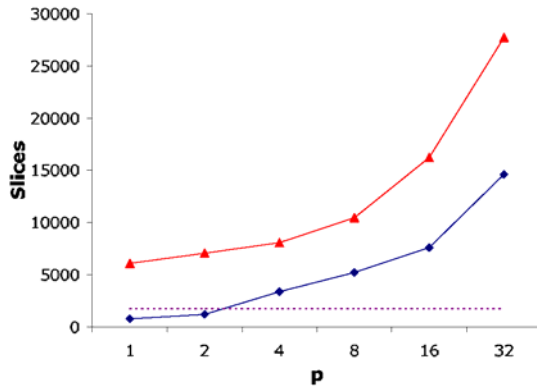
DFT₆₄



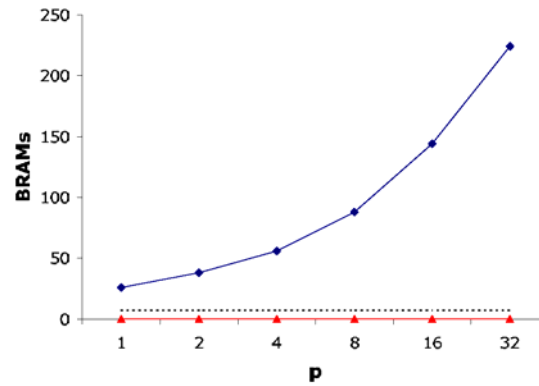
- ▲— Min BRAM
- ◆— Min Slice
- - - Xilinx

DFT_{1024} and DFT_{2048}

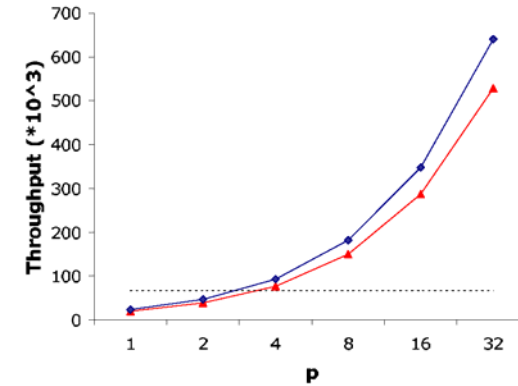
Number of Slices for DFT1024



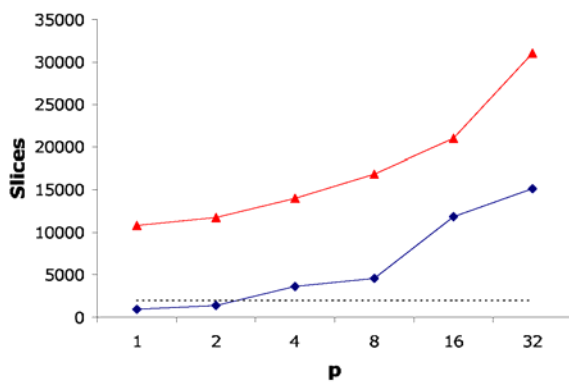
Number of BRAMs for DFT1024



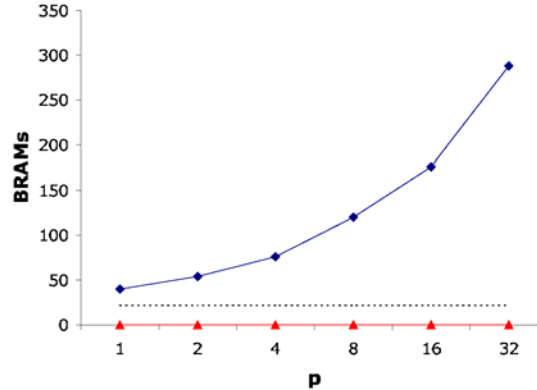
Throughput of DFT1024



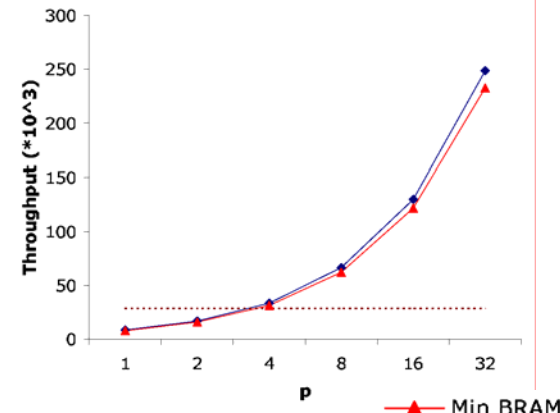
Number of Slices for DFT2048



Number of BRAMs for DFT2048



Throughput of DFT2048



▲ Min BRAM
◆ Min Slice
... Xilinx

Outline

- ◆ Introduction
- ◆ Formula-Driven Design Generation
- ◆ Microarchitecture Parameterization
- ◆ Resource Parameterization
- ◆ Experimental Results
- ◆ Conclusions

Related Work

- ◆ Kumhom, Johnson, Nagvajara, ASIC/SOC 2000
 - universal FFT processor microarchitecture based on processing elements interconnected by on-chip reconfigurable network
 - microarchitecture is scalable in the number of elements
 - supports both Cooley Tukey and Pease
- ◆ Choi, Scrofano, Prasanna, Jang, FPGA'2003
 - mapped radix-4 Cooley-Tukey algorithm onto $\log(n)/2$ DFT₄ primitives
 - scalable datapath between 1 element and 4 elements at a time
 - show energy and performance improvements from scaling
 - does not show same tradeoff point as Xilinx can be covered

Conclusions

- ◆ Parameterized IP Generator
 - easy to use
 - allows customization
- ◆ Prototype implementation of DFT generator
 - parameterized performance/cost tradeoff
 - parameterized resource usage preference
- ◆ Key results
 - generator is efficient, i.e., the Xilinx design point can be matched
 - customization allows advantage in a chosen dimension relative to Xilinx DFT cores